Recurrent Neural Networks

Why use an RNN?

- QUESTION: how would you build a neural network to predict the last amino acid of a peptide?
- **Consider:**
  - Previous amino acids important
  - Variable peptide sizes
  - padding not desirable
  - Want to use representations of an amino acid (encoding)

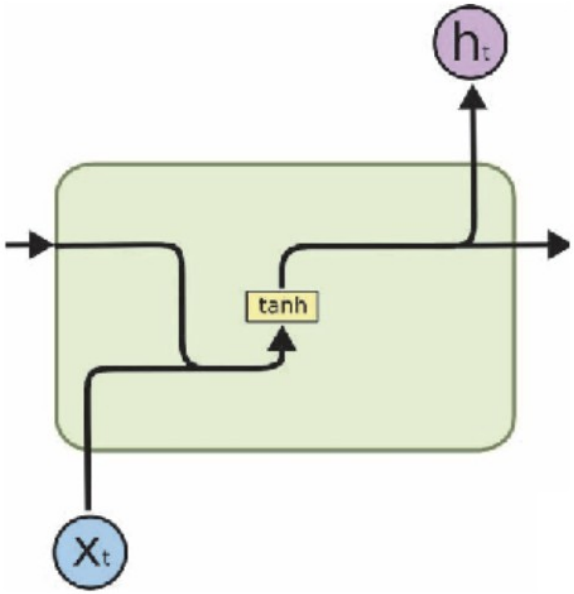
Why use an RNN?

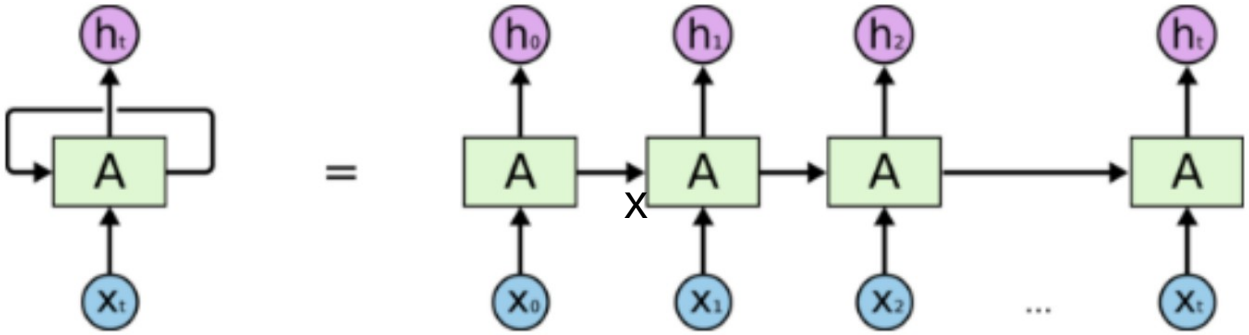
- Variable input / output length
  - But same input / output dimensions!
- Memory of previous states
- Encode sequences (future lecture (?))
- Note: RNN is an umbrella term

RNN key concepts

- Predict something based on previous states (e.g. last word of a sentence)
- Uses representation of previous state(s)
  - Add new information for each iteration
- Representation just another set of neurons
- Representation often called hidden

RNN Schematics





An unrolled recurrent neural network.

MIT lecture

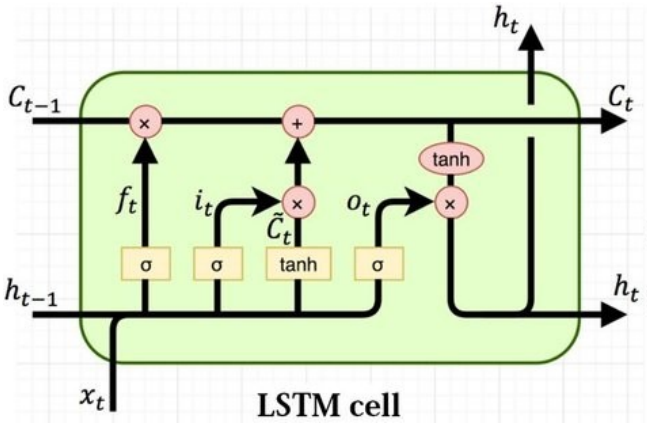
- https://www.youtube.com/watch?v=_h66BW-xNgk
- Why not feed-forward: 00:00 – 7:15
- RNN concept: 10:07 – 13:56
- Backprop: 13:56 – 17:42
- Vanishing gradient: 17:42 – 20:38

Visual representation of data flow

- https://youtu.be/8HyCNIVRbSU?t=187
- https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21
-

LSTM key concepts

- Long Short Term Memory RNN
- Selectively remove, use and change memory
  - Forget what is irrelevant, update with new information
  - Use only what is relevant



$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$
$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$
$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$
$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$
$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$
$$h_t = \tanh(C_t) * o_t$$

MIT lecture

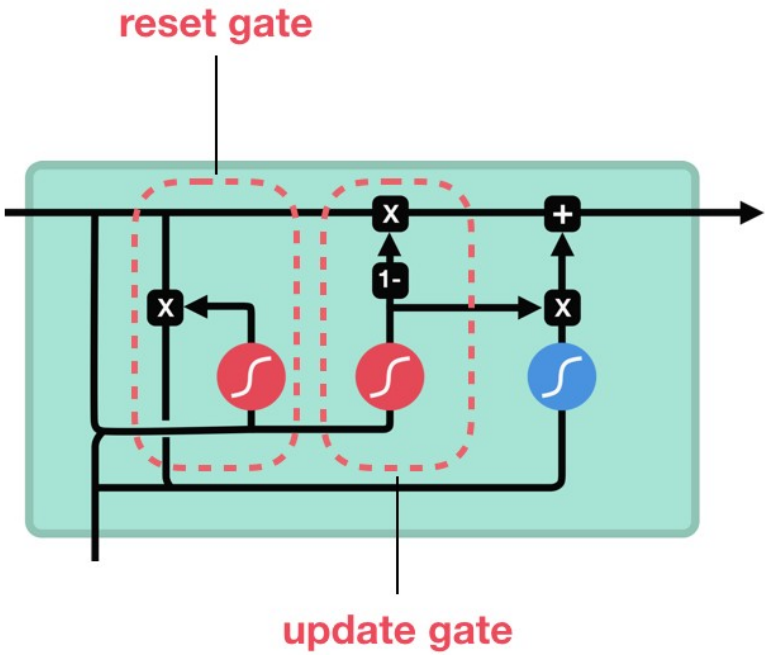
- https://youtu.be/_h66BW-xNgk?t=1228
- 30:30-end: applications
  - Extra if time allows for it

Visual representation of data flow

- https://youtu.be/8HyCNIVRbSU?t=175
- https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21
-

GRU key concepts

- Another RNN
- Forget and update gate
-



sigmoid

tanh

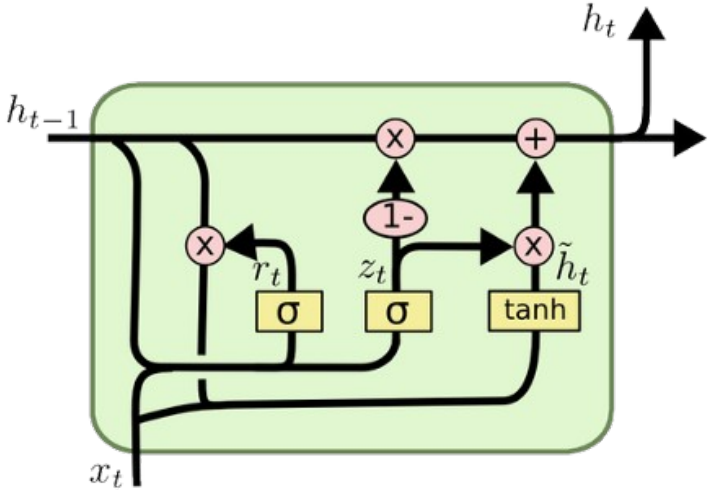
pointwise multiplication

pointwise addition

vector concatenation

GRU Lecture

- https://youtu.be/8Q582ng8Lxo?t=291
- Watch until 7:40



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

GRU vs LSTM

- GRU newer and less computation
- LSTM arguably slightly better
- No major differences as far as I know
- Try both

Further reading

Background / other explanations

- Understanding LSTM networks: http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Understanding GRU networks: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be
- Long Short-Term Memory: From Zero to Hero with PyTorch: https://blog.floydhub.com/long-short-term-memory-from-zero-to-hero-with-pytorch/

Interesting

- The Unreasonable Effectiveness of Recurrent Neural Networks: http://karpathy.github.io/2015/05/21/rnn-effectiveness/